View metada

WIRTSCHAFTS UNIVERSITÄT WIEN VIENNA UNIVERSITY OF ECONOMICS AND BUSINESS

EFMD EQUIS ACCREDITED

ePub^{WU} Institutional Repository

Gerhard Derflinger and Wolfgang Hörmann and Josef Leydold and Halis Sak

Efficient Numerical Inversion for Financial Simulations

Working Paper

Original Citation:
Derflinger, Gerhard and Hörmann, Wolfgang and Leydold, Josef and Sak, Halis (2009) Efficient Numerical Inversion for Financial Simulations. *Research Report Series / Department of Statistics and Mathematics*, 87. Department of Statistics and Mathematics, WU Vienna University of Economics and Business, Vienna.

This version is available at: http://epub.wu.ac.at/830/
Available in ePub^{WU}: June 2009

ePub^{WU}, the institutional repository of the WU Vienna University of Economics and Business, is provided by the University Library and the IT-Services. The aim is to enable open access to the scholarly output of the WU.

http://epub.wu.ac.at/

Efficient Numerical Inversion for Financial Simulations

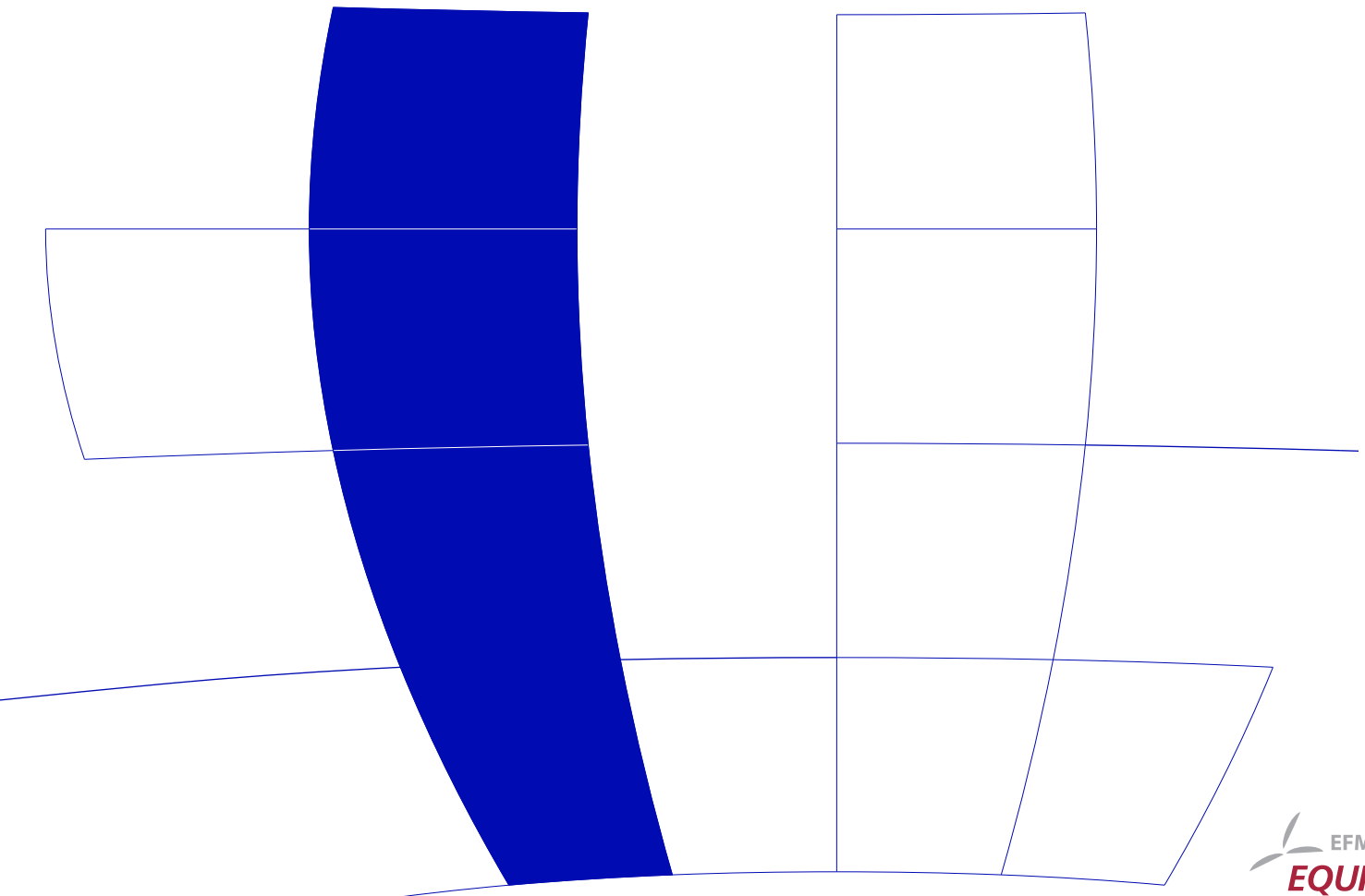
Gerhard Derflinger, Wolfgang Hörmann, Josef Leydold, Halis Sak

Department of Statistics and Mathematics
WU Wirtschaftsuniversität Wien

Research Report Series

Report 87
June 2009

http://statmath.wu.ac.at/



Efficient Numerical Inversion for Financial Simulations*

Gerhard Derflinger, Wolfgang Hörmann, Josef Leydold, and Halis Sak

Abstract Generating samples from generalized hyperbolic distributions and non-central chi-square distributions by inversion has become an important task for the simulation of recent models in finance in the framework of (quasi-) Monte Carlo. However, their distribution functions are quite expensive to evaluate and thus numerical methods like root finding algorithms are extremely slow. In this paper we demonstrate how our new method based on Newton interpolation and Gauss-Lobatto quadrature can be utilized for financial applications. Its fast marginal generation times make it competitive, even for situations where the parameters are not always constant.

1 Introduction

The evaluation of the quantile function of a given distribution is an inevitable task in the framework of Quasi-Monte Carlo methods (QMC) or for copula methods. Unfortunately, fast and accurate implementations of such functions are available only for a few of the important distributions, e.g., for the Gaussian distribution. Otherwise, one has to invert one's cumulative distribution function (CDF), for which it is usually easier to find a ready-to-use implementation (or at least a published

Gerhard Derflinger · Josef Leydold · Halis Sak
Department of Statistics and Mathematics, WU (Vienna University of Economics and Business), Augasse 2-6, A-1090 Vienna, Austria, e-mail: Gerhard.Derflinger@wu.ac.at — e-mail: Josef.Leydold@wu.ac.at — e-mail: Halis.Sak@wu.ac.at

Wolfgang Hörmann
Department of Industrial Engineering, Boğaziçi University, 34342 Bebek-Istanbul, Turkey, e-mail: hormannw@boun.edu.tr

* Paper published in: Pierre L'Ecuyer and Art B. Owen (eds.), *Monte Carlo and Quasi-Monte Carlo Methods 2008*, Springer-Verlag.
The original publication is available at www.springerlink.com.

algorithm). This procedure is called the *inversion method* in the random variate generation literature. Thus one has to apply some numerical root finding algorithms, usually Newton's method, variants of the secant method or interval bisection. Such an approach is even necessary for standard distributions having shape parameters. Then numerical inversion is combined with rough approximations to get starting points for recursive algorithms, see, e.g., the implementation of quantile functions for Gamma and t distributions in R [12].

For the *fixed parameter* case, i.e., when we have to draw (large) samples from the same distribution, a table based approach combined with interpolation is much faster. Possible implementations are proposed in [2, 6, 14]. Although the computation of the necessary coefficients during the setup can be quite expensive these methods are applicable to all distributions that fulfill some regularity conditions (like smooth and bounded density) and they have fast marginal generation times that hardly depend on the target distribution.

Recent developments of more realistic models for the dynamics of asset prices, interest or exchange rates lead to an increased application of less frequently used distributions. During the MCQMC'08 conference in Montreal we were impressed to see how many talks dealt with models that require the simulation of the variance gamma distribution or more generally of the generalized hyperbolic distribution. Also several authors report the good fit of the generalized hyperbolic distribution to daily stock-returns (see, e.g., [1] and [3]). Both at the conference and in the literature we observed that most researchers seemingly considered generating generalized hyperbolic random variates by inversion as too slow or even impossible. An exception was the talk by Tan [7] who used the algorithm described in [6]. He also mentioned that the setup took a long time. The slow setup is due to the fact that this algorithm requires the CDF which is extremely expensive for the generalized hyperbolic distribution. This talk motivated us to demonstrate the practical application of our recently proposed algorithm [5] to such distributions. It only requires the probability density function (PDF) of the target distribution and computes the CDF during the setup. The synergy of using interpolation together with numerical integration speeds up the setup for the generalized hyperbolic distribution by about hundred times compared to methods that are based on the direct evaluation of the CDF. Moreover, the algorithm allows to control the accuracy of the approximate inverse CDF.

A second numerically difficult distribution required in financial simulations is the non-central chi-square distribution. It is, e.g., required for simulating the increments of the well known Cox-Ingersoll-Ross model for the dynamics of short-term interest rates. Here the application of inversion algorithms is more difficult as the non-centrality parameters λ varies. Nevertheless, as the value of λ is close to 0 for all practically relevant choices of the parameters of that process, we were able to apply our algorithm for this simulation problem.

In this paper we explain the main idea of our newly proposed algorithm and discuss how an inaccurate evaluation of the PDF influences the error of the algorithm. We then develop the details necessary for its application to two simulation problems of quantitative finance requiring the generalized hyperbolic distribution with fixed

parameters and the non-central chi-square distribution. A ready-to-use implementation of our new black-box algorithm can be found as method PINV in our library UNU.RAN [9, 10].

All our experiments were performed in R as it provides a convenient platform for interactive computing. Densities for our target distributions are already available and the package *Runuran* [9] provides an interface to our C library. Of course our experiments can be conducted using C or any appropriate computing environment that provides an API to use a C library.

2 The Automatic Algorithm

Our algorithm has been designed as a black-box algorithm, i.e., the user has to provide a function that evaluates the PDF together with a "typical" point of the target distribution, and a maximal tolerated approximation error. As it is not tailored for a particular distribution family it works for distributions with smooth and bounded densities but requires some setup where the corresponding tables have to be computed. We only sketch the basic idea and refer the reader to [5] for all details (including the pseudo-code) and for arguments for the particular choice of the interpolation method and quadrature rule.

Measuring the Accuracy of Approximate Inversion

A main concern of any numerical inversion algorithm must be the control of the approximation error, i.e., the deviation of the approximate inverse CDF F_a^{-1} from the exact function F^{-1}. We are convinced that the *u-error* defined by

$$\varepsilon_u(u) = |u - F(F_a^{-1}(u))| \tag{1}$$

is well-suited for this task. In particular it can easily be computed during the setup and it can be interpreted with respect to the resolution of the underlying uniform pseudo-random number generator or low discrepancy set (see [5] for details). We therefore call the maximal tolerated u-error the *u-resolution* of the algorithm and denote it by ε_u in the sequel. We should mention here that the x-error, $|F^{-1}(u) - F_a^{-1}(u)|$, may be large in the tails of the target distribution. Hence our algorithms are not designed for calculating exact quantiles in the far tails of the distribution.

Newton's Interpolation Formula and Gauss-Lobatto Quadrature

For an interval $[b_l, b_r]$ we select a fixed number of points $b_l = x_0 < x_1 < \cdots < x_n = b_r$ and compute $u_i = F(x_i) = F(x_{i-1}) + \int_{x_{i-1}}^{x_i} f(x)\,dx$ recursively using $u_0 = 0$. The numeric integration is performed by means of Gauss-Lobatto quadrature with

5 points. The integration error is typically much smaller than the interpolation error and can be controlled using adaptive integration. We then construct a polynomial $P_n(x)$ of order n through the $n+1$ pairs (u_i, x_i), thus avoiding the evaluation of the inverse CDF F^{-1}. The coefficients of the polynomial are calculated using inverse Newton interpolation. Note that using numeric integration is often more stable than the direct use of an accurate implementation of the CDF due to loss of significant digits in the right tail. The interpolation error can be computed during the setup. It is possible to search for the maximal error over $[F(b_l), F(b_r)]$, but we suggest to use a much cheaper heuristic, that estimates the location of the maximal error using the roots of Chebyshev polynomials. The intervals $[b_l, b_r]$ are constructed sequentially from left to right in the setup. The length of every interval is shortened till the estimated u-error is slightly smaller than the required u-resolution.

Cut-off Points for the Domain

The interpolation does not work for densities where the inverse CDF becomes too steep. In particular this happens in the tails of distributions with unbounded domains. Thus we have to find the computational relevant part of the domain, i.e., we have to cut off the tails such that the probability of either tail is negligible, say about 5% of the given u-resolution ε_u. Thus it does not increase the u-error significantly. We approximate the tail of the distribution by a function of the form x^{-c} fitted to the tail in a starting point x_0 and take the cut-off value of that approximation.

3 Considerations for Approximate Densities

The error control of our algorithm assumes that the density can be evaluated precisely. However, in practice we only have a (albeit accurate) approximate PDF $f_e(x)$ available. The pointwise error of the corresponding approximate CDF F_e is bounded by the L_1-error of f_e which is defined by

$$|F(x) - F_e(x)| \le L_1\text{-error} = \varepsilon_1 = \int_{-\infty}^{\infty} |f(x) - f_e(x)|\, dx\,.$$

Hence, the total resulting maximal u-error of the approximation F_a^{-1} is bounded by $\varepsilon_u + \varepsilon_1$. Thus, if we can obtain an upper bound $\tilde{\varepsilon}_1$ for the L_1-error, we can reduce the parameter ε_u of our algorithm by $\tilde{\varepsilon}_1$ to get an algorithm that is guaranteed to have the required u-resolution. (Of course this requires that ε_1 is sufficiently small.)

The L_1-error can be estimated by means of high precision arithmetic (we used *Mathematica* [15] for this task). We compared our implementation of a particular PDF with one that computes 30 significant decimal digits at 500,000 equidistributed points and calculated the L_1-error.

4 The Generalized Hyperbolic Distribution

The generalized hyperbolic distribution is considered to be a more realistic albeit numerically difficult model for the increment of financial processes or the marginal distributions of portfolio risk. However, for both, the generation of a variance gamma process using QMC as well as the generation of the marginal distribution from (e.g.) a t-copula the inversion method is inevitable. We therefore discuss here the application of our numerical inversion algorithm to that important distribution family.

Our tests were performed with the parameter values estimated for four different German stocks in [11]. We first estimated the L_1-errors of our implementation of the PDF (using the R library function for the modified Bessel function of third kind) which was always below 10^{-15}, i.e., close to machine precision of the double format of the IEEE floating point standard.

Our numerical inversion algorithm works for all parameter sets. A u-resolution of 10^{-12} and an order $n = 5$ for the polynomial interpolation requires 140 intervals and the setup was executed in less than 0.2 seconds. This is about 100 times faster than using the inversion algorithm of [6] that requires the CDF instead of the PDF. The observed u-error remained always below the required u-resolution (we used R package *ghyp* [4] as an independent implementation of the CDF of the generalized hyperbolic distribution). The marginal execution time is very fast, less than 0.2 seconds for generating one million variates. So including the setup we are able to sample 10^7 variates of a generalized hyperbolic distribution in less than 2 seconds which is quite fast. Compared to inversion using the quantile function of the *ghyp* package, that requires 35 seconds to generate 1000 variates, the speed up factor is 10^5 which is impressive.

This means that for the realistic return model using the t-copula with generalized hyperbolic marginals our new numeric inversion allows to obtain precise estimates of value at risk or expected shortfall in acceptable time. In that model the return distribution of the d assets of the portfolio has different parameters. As the generation of a return vector requires the evaluation of the inverse CDF of each of these d parameter sets it is therefore necessary to start the simulation with calculating d sets of constants that are stored in d different "generator objects". Thus our numeric inversion algorithm can be used in this semi-varying parameter situation, also values of d around 50 or 100 are no problem.

5 The Non-central Chi-square distribution

The density of the non-central chi-square distribution is difficult to evaluate. Its R implementation uses the representation of the non-central chi-square distribution as a Poisson mixture of central chi-square distributions [8, p. 436]. The density is bounded for $\nu \geq 2$ degrees of freedom. Our numerical estimation shows that the L_1-error of the R implementation is never larger than 10^{-14}. So our algorithm can be applied to get a fast and sufficiently accurate inversion algorithm for the non-

central chi-square distribution. An u-resolution of 10^{-12} and an order $n = 5$ for the polynomial interpolation requires a bit more than 200 intervals for all parameter values of our experiments. The setup together with the generation of 10^6 variates by inversion took about 0.27 seconds and was thus faster than generating 100 variates by inversion using the built in R function `qchisq`. So here the algorithm reached a speed-up factor above 10^4. Note that our inversion algorithm is also about 30 percent faster than using the random variate generation function `rchisq` that is using the mixture representation of the non-central chi-square distribution and is therefore not an inversion algorithm.

Random Variate Generation for the CIR Model

The Cox-Ingersoll-Ross or CIR model is a well known single factor interest rate model defined by the stochastic differential equation

$$dr_t = \kappa(\theta - r_t)dt + \sigma\sqrt{r_t}d\omega_t \ .$$

The increments of the process r_t for a time jump of size T follow a multiple of the non-central chi-square distribution with $\nu = 4\kappa\theta/\sigma^2$ degrees of freedom and non-centrality parameter $\lambda = (1-\exp(-\kappa T))\sigma^2 r_t \exp(-\kappa T)/\kappa$.

So path simulation requires only the generation of non-central chi-square variates. For fixed process parameters, ν is fixed, whereas λ depends on the current value r_t and thus changes for every call. The changing parameter situation seems to be an obstacle for using a table based inversion algorithm. But a closer look at the formula for λ shows that for sensible process parameters, λ is always close to 0. To check this observation we investigated a total of 24 parameter sets estimated in [13] for the overnight rates of European interbank, of London interbank for Euro, and the overnight rates for Poland, Slovakia, Hungary and the Czech Republic. The parameters were estimated for the four quarters of 2003 separately. Assuming that r_t is not larger than $10\,\theta$ and trying values of T between 0.001 and 1 it turns out that we always had $\lambda < 0.011$. As ν is fixed and the range of possible λ values is so small, we can use our inversion algorithm for this varying parameter situation. The simplest approach is to run the setup of the algorithm for the parameters $(\nu, \lambda = 0)$ and for $(\nu, \lambda = 0.011)$. To make inversion for an arbitrary value of λ in $(0, 0.011)$ we calculate both x-values, that for $(\nu, \lambda = 0)$ and $(\nu, \lambda = 0.011)$, and then use linear interpolation in λ. It is enough to use $\varepsilon_u = 10^{-10}$ such that the result of the simple linear interpolation in λ has an u-error smaller than 10^{-7} for $\nu \geq 6$ and smaller than $3.5 \cdot 10^{-7}$ for $\nu \geq 3$.

Remark 1. As λ is so close to zero we first thought that it could be enough to use $\lambda = 0$ (i.e., inversion with the ordinary chi-square distribution) as approximation. But that simple approximation leads, e.g., for $\nu = 4$ and $\lambda = 0.005$ to a u-error larger than 10^{-3} which is certainly not acceptable.

If smaller u-errors or larger λ values are required one may use quadratic interpolation in λ. For example for $\lambda \leq 0.1$ we run the setup and store the respective

```
library(Runuran)   ## load library

qnccsi <- function(u, nu, lambda) {
   ## approx. inverse CDF of non-central chi-square distribution.
   ##   u      ... probabilities (vector of length n)
   ##   nu     ... degrees of freedoms (numeric)
   ##   lambda ... non-central values (vector of lenght n)
   ## quadratic interpolation for lambda.
   ## (default) uresolution=1e-10 for generator PINV

   # maximum of non-centrality parameter
   maxlambda <- max(lambda)
   # "typical" point of distributions
   xc <- 0.5*nu
   # generators for distributions
   myf0 <- function(x) dchisq(x,df=nu,ncp=0)
   gen0 <- pinv.new(pdf=myf0,lb=0,ub=Inf,center=xc)
   myf1 <- function(x) dchisq(x,df=nu,ncp=maxlambda*0.5)
   gen1 <- pinv.new(pdf=myf1,lb=0,ub=Inf,center=xc)
   myf2 <- function(x) dchisq(x,df=nu,ncp=maxlambda)
   gen2 <- pinv.new(pdf=myf2,lb=0,ub=Inf,center=xc)
   # generate points from these distributions
   x0<-uq(gen0,u); x1<-uq(gen1,u); x2<-uq(gen2,u)
   # interpolate for particular non-centrality parameters
   lam <- lambda*2/maxlambda-1
   x <- 0.5*(((x0+x2)*lam+(x2-x0))*lam)+x1*(1-lam*lam)
   # return random sample
   x
}

## draw a random sample with randomly selected lambda values
x <- qnccsi(u=runif(1e6),nu=20,lambda=runif(1e6)*0.1)
```

Fig. 1 R code for approximate inverse CDF of a non-central chi-square distribution with `nu` degrees of freedom (fixed) and varying non-centrality parameter `lambda`. (This code requires R version 2.8.1 or later since otherwise `dchisq` hangs.)

constants for $\lambda_0 = 0$, $\lambda_1 = 0.05$, and $\lambda_2 = 0.1$. For an arbitrary $\lambda \leq 0.1$ and u we evaluate the inverse CDF and calculate $x_i = F^{-1}(u, \lambda_i)$ for $i = 0, 1, 2$ using the three stored tables. The final result is then obtained using quadratic interpolation of the three pairs (λ_i, x_i). Figure 1 lists an R function that implements such an approach. Using quadratic interpolation in λ we observed an u-error smaller than $1.5 \cdot 10^{-7}$ for $\nu \geq 3$, $\lambda \leq 0.1$ and an u-error smaller than $2.3 \cdot 10^{-8}$ for $\nu \geq 6$, $\lambda \leq 0.1$. The u-error is more than 100 times smaller than when using linear interpolation for those parameter values. Of course the quadratic interpolation and the evaluation of three quantiles takes some time. Generating 10^6 variates for varying λ-values with the R code in Fig. 1 takes about 0.7 seconds. Compared to the 2.5 seconds it takes to generate 1000 of the same variates with the built-in quantile function of R we can still observe a speed-up factor above 3000.

References

1. Aas, K., Haff, I.H.: The generalized hyperbolic skew Student's t-distribution. Journal of Financial Econometrics **4**(2), 275–309 (2006)
2. Ahrens, J.H., Kohrt, K.D.: Computer methods for efficient sampling from largely arbitrary statistical distributions. Computing **26**, 19–31 (1981)
3. Behr, A., Pötter, U.: Alternatives to the normal model of stock returns: Gaussian mixture, generalised logF and generalised hyperbolic models. Annals of Finance **5**(1), 49–68 (2009). DOI 10.1007/s10436-007-0089-8
4. Breymann, W., Lüthi, D.: ghyp: A package on generalized hyperbolic distributions. Tech. rep., Institute of Data Analysis and Process Design (2008). `http://cran.r-project.org/`
5. Derflinger, G., Hörmann, W., Leydold, J.: Numerical inversion when only the density function is known. Preprint Series of the Department of Statistics and Mathematics 78, Wirtschaftsuniversität Wien, Augasse 2–6, A-1090 Wien, Austria (2008). `http://epub.wu-wien.ac.at/english/`
6. Hörmann, W., Leydold, J.: Continuous random variate generation by fast numerical inversion. ACM Transactions on Modeling and Computer Simulation **13**(4), 347–362 (2003)
7. Imai, J., Tan, K.S.: An enhanced quasi-Monte Carlo method for simulating generalized hyperbolic Levy process (2008). Talk at the MCQMC08 in Montreal
8. Johnson, N.L., Kotz, S., Balakrishnan, N.: Continuous Univariate Distributions, vol. 2, 2nd edn. Wiley, New York (1995)
9. Leydold, J., Hörmann, W.: Runuran – R interface to the UNU.RAN random variate generators, Version 0.8. Department of Statistics and Mathematics, WU Wien, A-1090 Wien, Austria (2008). `http://cran.r-project.org/`
10. Leydold, J., Hörmann, W.: UNU.RAN – A Library for Non-Uniform Universal Random Variate Generation, Version 1.3. Department of Statistics and Mathematics, WU Wien, A-1090 Wien, Austria (2008). `http://statmath.wu-wien.ac.at/unuran/`
11. Prause, K.: Modelling financial data using generalized hyperbolic distributions. Tech. rep., University of Freiburg (1997)
12. R Development Core Team: R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, Vienna, Austria (2008). URL `http://www.R-project.org`. ISBN 3-900051-07-0
13. Ševčovič, D., Urbánová Csajková, A.: On a two-pahse minmax method for parameter estimation of the cox, ingersoll, and ross interest rate model. Central European Journal of Operations Research **13**, 169–188 (2005)
14. Ulrich, G., Watson, L.: A method for computer generation of variates from arbitrary continuous distributions. SIAM J. Sci. Statist. Comput. **8**, 185–197 (1987)
15. Wolfram Research, Inc.: Mathematica, Version 6.0. Champaign, IL (2007)